Advanced Java Programming

Assignment 1

1. Write a `VendingMachine` class to simulate interactions with a vending machine. It should allow users to deposit coins with `public void deposit(int cents)`, and then to purchase a snack with `public boolean purchase(String snack)`, which should return `true` if the snack was successfully purchased and `false` if not enough money was deposited. Finally, it should support `public int getChange()` to have the machine return all money deposited. When writing this program, be sure to consider invalid user inputs and mark your states `private`.

   Note that the `VendingMachine` class needs to keep track of snacks available and their prices. This field should be `static`, so prices are constant across all `VendingMachine` objects. How you decide to keep track of this information is up to you. A couple possibilities include:

   a. Using parallel arrays to represent snack names and prices
   b. Creating a `Snack` object with `name` and `price` fields, and maintaining an array of `Snack`s
   c. If you are already familiar with Java's `HashMap` class, you can create a mapping of names to prices.

   You should also implement `public static int getPrice(String snack)` for user convenience.

2. Write a `Fraction` class to represent a fractional number. It should have two states, `numerator` and `denominator`, both of which should be marked `private` and should have accessor methods. You should have two constructors: one should accept two integers as the numerator and denominator, and the other should accept a single integer to represent a whole number. The fraction should be stored in its most reduced form. You should support the methods `public Fraction add(Fraction other)` that adds two `Fraction` objects and returns a new `Fraction` object representing the result, and should similarly support `public Fraction subtract(Fraction other)`, `public Fraction multiply(Fraction other)`, and `public Fraction divide(Fraction other)`. All of these methods should return `Fraction` objects that have been reduced as much as possible. You should also have a `public Fraction getReciprocal()` method that returns a Fraction that represents the reciprocal of this fraction.

Finally, you should have a `public double toDouble()` method that converts the fraction into a floating point value.

3. Write a `Timer` class to time how long a piece of code takes to run. It should have a method `public void start()` that should be invoked when you want to start timing, and a method `public void stop()` that should be invoked when you want to stop timing. The method `public double getElapsedTime()` should return the number of seconds the code took to run. If `stop()` is invoked before `start()` or multiple times in a row, it should have no effect. If `start()` is invoked multiple times in a row, it should have no effect. The time between invocations of `start()` and `stop()` should accumulate, so if between the first `start()` and `stop()` 1 second elapsed and between the second `start()` and `stop()` 2 seconds elapsed, then `getElapsedTime()` should return 3 seconds. The timer should be able to be reset with `public void reset()`, in which case the accumulated time will be dropped. If `reset()` is invoked while the `Timer` is running, it should have no effect.

   In order to get this program to work, you will need to make use of the `System.currentTimeMillis()` method, which returns a `long` representation of the current time in milliseconds.